Assignment 2

STAT 4620/5620 Winter 2025

Submission Instructions

- Follow the instructions in Question 1 to create an R package with version control tracked by Git and upload it to GitHub. Include a link to the GitHub repository at the end of Question 1.
- Modify this Quarto document to answer the questions and upload your Quarto file and rendered PDF file to Brightspace.

Question 1 (14 points)

- This question will guide you through the creation of an R package that can be used to increase the reproducibility and shareability of your data analyses.

1. Install the `devtools` package.
2. In R Studio click the "Create a Project → New Directory → R Package using devtools". Use the directory name "A2NAME", replacing NAME with your name. Feel free to pick any folder for "Create project as a subdirectory of:", a good option is your Data Analysis STAT4620/5620 folder for the course, if you have one. Click "Create Project". R Studio should automatically switch to the newly created project, switch to it yourself if not.
3. Edit the package DESCRIPTION file to write your name in the author field.
4. Start using version control for your package.
    1. Install git if you have not already. Unless you know what you're doing, when the git installer asks you to choose the default editor used by git, switch it from Vim to something more familiar like Notepad. Use the defaults for the rest of the options. Restart Rstudio.
    2. Find the "Terminal" pane in R Studio and type in the following commands, substituting in your own name and e-mail:

    - `git config --global user.name 'Ethan Lawler'`
    - `git config --global user.email 'lawlerem@dal.ca'`
   3. In R Studio press “Tools → Version Control → Project Setup…”. A pop-up window should show up with the Git/SVN menu selected. Change “Version control system…” to “Git”, and say “Yes” when it asks you if you want to initialize a new git repository for this project, restarting R Studio if necessary.
   4. Find the “Git” panel in R Studio. This panel lists the files and folder in your package and gives you some information about each of them. Check the box for each file under the “Staged” column and make sure the “Status” column changes to a green A that says “Added” if you hover your cursor over it.
   5. When all files are added press the “Commit” button. In the pop-up window write “Initial commit” in the “Commit message” box. Press the “Commit” button, then feel free to close all the windows except for your main R Studio window.
   6. Still in the “Git” panel, press the “History” button. You should see a pop-up window containing a line saying “Initial commit” along with your name, e-mail address, and date that you completed the previous step. Close the pop-up window.

5. In the R folder of your package create an R file named `mean.R`. The `mean.R` file should contain a function `mymean` that takes a numeric vector as input and returns the mean of the vector. Make sure your `mymean` function is fully documented using tags, including a title, the name and description of the input vector, and a description of what the function returns. Make sure your documentation includes a tag to export your function. Run `devtools::document()` in the “Console” panel to create help pages from your documentation. You should notice that NAMESPACE, R/, and man/ now appear in the “Git” panel. Add them by checking the boxed under the “Staged” column, the NAMESPACE status should change to a blue M to show that the NAMESPACE file was already part of the git repository but has been changed since the previous commit. Commit the files and folders using the commit message “Added and documented a function to compute a mean.”

6. Load the *cars* data.frame included in R then run `usethis::use_data(cars)` to add the *cars* dataset to your package. You should notice that a “data” folder has been added to your package. Add a *data.R* file to the R folder of your package and document the cars dataset there. Run `devtools::document()` to create help pages from your documentation. Add all of your new files and create a commit using the message “Added and documented a copy of the cars dataset.”

7. Create a package vignette with the code `usethis::use_vignette("A2NAME.qmd")`, replacing NAME with your name. You can use a package vignette to write a reproducible data analysis that uses the function you write in your R/ folder and the data you add to your package. For now you can leave the vignette empty. Add all the files to git and commit them with the message “Completed Question 1!”.

Now we'll set up your GitHub account and post your R package on GitHub. This step will be a bit complicated but you only ever need to do this set-up once. We will - Create a github account and an automatic password called a personal access token. - Create an ssh key that lets you move things from your computer to GitHub. - Create a new repository on GitHub then upload your package to GitHub.

8. Go to github.com and create a GitHub account (or use an existing one). To upload to GitHub you need a Personal Access Token, go to this link and follow the instructions for "Creating a personal access token (classic)". When you get to step 8 and are asked to select the scopes you'd like to grant, make sure the "repo" box is checked. Click the clipboard icon next to your newly created personal access token to copy it. In Rstudio, install the "gitcreds" package the run `gitcreds::gitcreds_set()`. When it asks you to "enter password or token:" paste in the copied personal access token and press enter. Restart RStudio.
9. Back on GitHub, go the the "Your repositories" page and click the "New" button to create a repository for your R package. The repository name should have the same "A2NAME" name as your package. Keep the default option to make it a public repository and click the "Create repository" button.
10. In the "Quick setup" box, make sure the "HTTPS" button is selected. Copy the first two lines of the box under "…or push an existing repository from the command line". The first line should contain a url for you new repository that starts with "https://". Open the Terminal panel in Rstudio, paste in those contents and press enter. In the Git panel of Rstudio, press the "Push" button to upload your files to your newly created GitHub repository.
11. Go to your package repository on Github and check to see that all of the files in your package are now uploaded to GitHub. Copy the url for your GitHub repository and paste it below.

GitHub Repository URL: https://github.com/MHajatiDAL/A2Hajati

Question 2 (4 points)

- Explain how the Akaike information criterion (AIC) is computed for a generalized linear model and how it is commonly used for model or variable selection purposed. Be sure to describe the two competing goals that AIC tries to find a good balance for. (250 words)

  AIC is computed using this formula:

  $$AIC = -2 \log L + 2k$$

  Where the log L is the log likelihood of the fitted model and k is the number of estimated parameters. For a GLM, the log-likelihood depends on the assumed distribution and the estimated parameters. so for more complex models usually give a yield a higher likelihood (log L), but this means that there is the cost of additional parameters (k), which increases the AIC value. You want to pick the model with lowest AIC value because it shows that it has a good balance of fitting the data and not being too complex with too many parameters. This quality makes the AIC a good tool to compare the models relatively but does not speak to the actual ability of any of the models to make good or bad predictions. AIC provides a measure of a models relative quality by balancing model fit and model simplicity. Models with more parameters generally fit the data better, but Overly complex models risk overfitting, so they look at the noise instead of the actual pattern, but the penalty term (2k) discourages this. So you need the balance.

Question 3 (4 points)

- Residual checking for GLMs is not always as straightforward as it is for linear models, and the problems are particularly acute in the case of binary responses. Explain why (100 words)

GLM usually involve non gaussian error distribution and the link function transforms the response variable making the process more complicated. For binary case, the issue is that the residual can either be 0 or 1 so its discrete instead of continous, meaning that the residual plots are less informative than they normally are. usually residuals are continous making them more informative but here they are discrete and do not follow normality. So this makes them really hard.

Question 4 (16 points)

- We are interested in a study concerning lung function in patients with cystic fibrosis (Altman 1991, p.338). Install the **ISwR** package and load the *cystfibr* dataset provided in that package.

  1. Fit a model relating maximum expiratory pressure (pemax) to the explanatory variables contained in the dataset. Describe the steps you take including fitting an initial model, residual analysis, and changes to your initial model if needed. Make sure you include the reasons why you make certain modelling choices.
  2. Interpret results for the sex variable.
  3. Try using the *step* function and interpret the results.
  4. What can you reasonably conclude from your analysis?

```r
if( !requireNamespace("ISwR", quietly = TRUE) ) install.packages("ISwR")
library(ISwR)
```

```
Warning: package 'ISwR' was built under R version 4.3.3
```

```r
data(cystfibr)

str(cystfibr)
```

```
'data.frame':   25 obs. of  10 variables:
 $ age   : int  7 7 8 8 8 9 11 12 12 13 ...
 $ sex   : int  0 1 0 1 0 0 1 1 0 1 ...
 $ height: int  109 112 124 125 127 130 139 150 146 155 ...
 $ weight: num  13.1 12.9 14.1 16.2 21.5 17.5 30.7 28.4 25.1 31.5 ...
 $ bmp   : int  68 65 64 67 93 68 89 69 67 68 ...
 $ fev1  : int  32 19 22 41 52 44 28 18 24 23 ...
 $ rv    : int  258 449 441 234 202 308 305 369 312 413 ...
 $ frc   : int  183 245 268 146 131 155 179 198 194 225 ...
 $ tlc   : int  137 134 147 124 104 118 119 103 128 136 ...
 $ pemax : int  95 85 100 85 95 80 65 110 70 95 ...
```

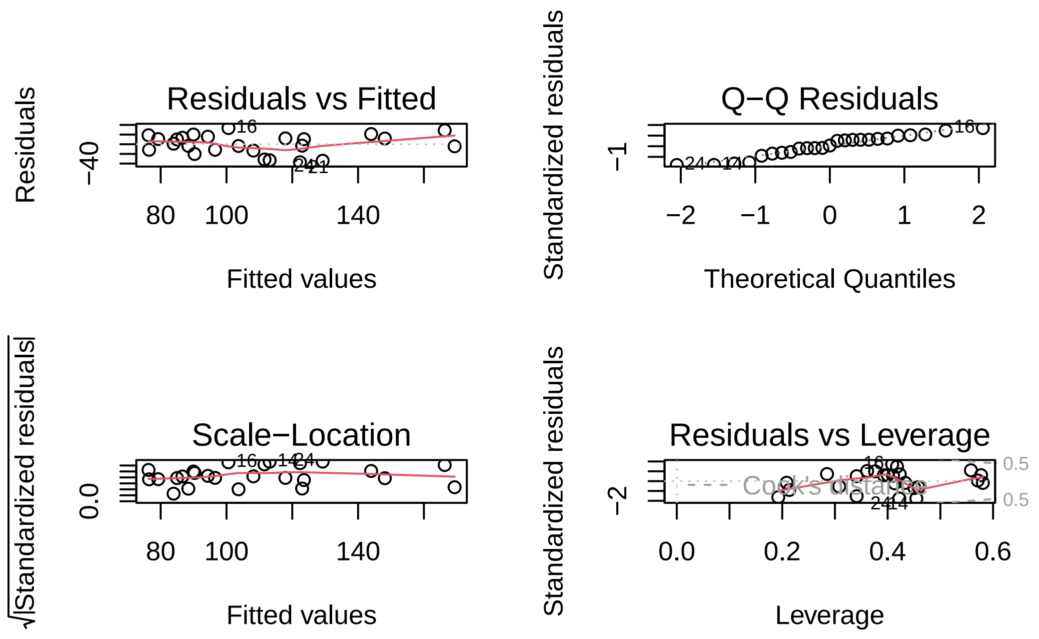
```r
summary(cystfibr)
```

```
      age             sex            height          weight           bmp       
 Min.   : 7.00   Min.   :0.00   Min.   :109.0   Min.   :12.9   Min.   :64.00  
 1st Qu.:11.00   1st Qu.:0.00   1st Qu.:139.0   1st Qu.:25.1   1st Qu.:68.00  
 Median :14.00   Median :0.00   Median :156.0   Median :37.2   Median :71.00  
 Mean   :14.48   Mean   :0.44   Mean   :152.8   Mean   :38.4   Mean   :78.28  
 3rd Qu.:17.00   3rd Qu.:1.00   3rd Qu.:174.0   3rd Qu.:51.1   3rd Qu.:90.00  
 Max.   :23.00   Max.   :1.00   Max.   :180.0   Max.   :73.8   Max.   :97.00  
      fev1             rv             frc             tlc          pemax      
 Min.   :18.00   Min.   :158.0   Min.   :104.0   Min.   : 81   Min.   : 65.0  
 1st Qu.:26.00   1st Qu.:188.0   1st Qu.:127.0   1st Qu.:101   1st Qu.: 85.0  
 Median :33.00   Median :225.0   Median :139.0   Median :113   Median : 95.0  
 Mean   :34.72   Mean   :255.2   Mean   :155.4   Mean   :114   Mean   :109.1  
 3rd Qu.:44.00   3rd Qu.:305.0   3rd Qu.:183.0   3rd Qu.:128   3rd Qu.:130.0  
 Max.   :57.00   Max.   :449.0   Max.   :268.0   Max.   :147   Max.   :195.0  
```

```r
# going with a linear to see if we are lucky
model1 <- lm(pemax ~ ., data = cystfibr)

summary(model1)
```

```
Call:
lm(formula = pemax ~ ., data = cystfibr)

Residuals:
    Min      1Q  Median      3Q     Max
-37.338 -11.532   1.081  13.386  33.405

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 176.0582   225.8912   0.779    0.448
age          -2.5420     4.8017  -0.529    0.604
sex          -3.7368    15.4598  -0.242    0.812
height       -0.4463     0.9034  -0.494    0.628
weight        2.9928     2.0080   1.490    0.157
bmp          -1.7449     1.1552  -1.510    0.152
fev1          1.0807     1.0809   1.000    0.333
rv            0.1970     0.1962   1.004    0.331
frc          -0.3084     0.4924  -0.626    0.540
tlc           0.1886     0.4997   0.377    0.711

Residual standard error: 25.47 on 15 degrees of freedom
Multiple R-squared:  0.6373,    Adjusted R-squared:  0.4197
F-statistic: 2.929 on 9 and 15 DF,  p-value: 0.03195
```

```r
# from the summary i can see that sex, age, height, weight, FEV1, RV, FRC, TLC
# have high P values so fail the statistical test

# plot to see if I can get anything from it
par(mfrow = c(2, 2))
plot(model1)
```



```
# observations 16 and 24 seem to be have a lot leverage so gonna check them
cystfibr[c(16, 24), ]
```

```
   age sex height weight bmp fev1  rv frc tlc pemax
16  17   1    153   34.8  70   29 204 118 120   134
24  23   0    175   51.1  71   33 224 131 113    95
```

```
# decided to delete them since 16 had short height and low weight
# compared to average and 24 was taller and heavier than average
cystfibr_filtered <- cystfibr[-c(16, 24), ]

# doing the model again with the new data
model_filtered <- lm(pemax ~ ., data = cystfibr_filtered)

# getting the summary again
summary(model_filtered)
```

```
Call:
lm(formula = pemax ~ ., data = cystfibr_filtered)

Residuals:
    Min      1Q  Median      3Q     Max 
-40.750  -8.483   5.730  12.966  21.326 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 149.47058  223.76642   0.668    0.516
age          -1.15692    6.07780  -0.190    0.852
sex          -9.99319   14.82852  -0.674    0.512
height       -0.40315    0.82280  -0.490    0.632
weight        2.74687    2.23357   1.230    0.241
bmp          -1.70542    1.12973  -1.510    0.155
fev1          1.17515    0.99876   1.177    0.260
rv            0.18718    0.18137   1.032    0.321
frc          -0.15868    0.45210  -0.351    0.731
tlc           0.05841    0.54954   0.106    0.917

Residual standard error: 23.12 on 13 degrees of freedom
Multiple R-squared:  0.7328,    Adjusted R-squared:  0.5479 
F-statistic: 3.962 on 9 and 13 DF,  p-value: 0.01256
```

```r
# in this one we have a higher R^2 meaning better fit
# a lower residual standard error again meaning better
# and a lower P value so our model is more statistically significant
# from this I can say that deleting 16 and 24 was the right call
# but i did this in a linear model, does that mean in other models
# 16 and 24 might actually be good or does this mean that
# 16 and 24 are bad across the board for all models?

# doing a stepwise selection
model_step <- step(model_filtered, direction = "both")
```

```
Start:  AIC=151.35
pemax ~ age + sex + height + weight + bmp + fev1 + rv + frc + 
    tlc

         Df Sum of Sq    RSS    AIC
- tlc     1      6.04 6954.9 149.37
- age     1     19.37 6968.3 149.41
- frc     1     65.85 7014.7 149.57
- height  1    128.33 7077.2 149.77
- sex     1    242.76 7191.7 150.14
- rv      1    569.34 7518.2 151.16
```

```
<none>                6948.9 151.35
- fev1    1    740.00 7688.9 151.68
- weight  1    808.44 7757.3 151.88
- bmp     1   1218.12 8167.0 153.06

Step:  AIC=149.37
pemax ~ age + sex + height + weight + bmp + fev1 + rv + frc

         Df Sum of Sq    RSS    AIC
- age     1     49.35 7004.3 147.53
- frc     1     60.49 7015.4 147.57
- height  1    155.85 7110.8 147.88
- sex     1    291.03 7246.0 148.31
- rv      1    567.09 7522.0 149.17
<none>                6954.9 149.37
- fev1    1    747.18 7702.1 149.72
+ tlc     1      6.04 6948.9 151.35
- weight  1   1402.15 8357.1 151.59
- bmp     1   1877.99 8832.9 152.87

Step:  AIC=147.53
pemax ~ sex + height + weight + bmp + fev1 + rv + frc

         Df Sum of Sq    RSS    AIC
- frc     1     22.17 7026.5 145.60
- height  1    152.83 7157.1 146.03
- sex     1    244.75 7249.0 146.32
- rv      1    522.72 7527.0 147.19
<none>                7004.3 147.53
- fev1    1   1145.35 8149.6 149.01
+ age     1     49.35 6954.9 149.37
+ tlc     1     36.02 6968.3 149.41
- bmp     1   2097.34 9101.6 151.56
- weight  1   2795.41 9799.7 153.26

Step:  AIC=145.6
pemax ~ sex + height + weight + bmp + fev1 + rv

         Df Sum of Sq    RSS    AIC
- height  1    131.73 7158.2 144.03
- sex     1    232.47 7258.9 144.35
<none>                7026.5 145.60
- rv      1   1136.49 8162.9 147.05
```

```
+ frc     1     22.17 7004.3 147.53
+ age     1     11.03 7015.4 147.57
+ tlc     1      8.23 7018.2 147.58
- fev1    1   2035.71 9062.2 149.46
- bmp     1   2569.37 9595.8 150.77
- weight  1   2773.26 9799.7 151.26

Step:  AIC=144.03
pemax ~ sex + weight + bmp + fev1 + rv

         Df Sum of Sq     RSS    AIC
- sex     1     219.8  7377.9 142.73
<none>                 7158.2 144.03
+ height  1     131.7  7026.5 145.60
+ tlc     1      43.6  7114.6 145.89
+ age     1      25.6  7132.6 145.95
+ frc     1       1.1  7157.1 146.03
- rv      1    1536.2  8694.4 146.50
- fev1    1    2375.3  9533.5 148.62
- bmp     1    2946.2 10104.3 149.96
- weight  1   12518.0 19676.2 165.29

Step:  AIC=142.73
pemax ~ weight + bmp + fev1 + rv

         Df Sum of Sq     RSS    AIC
<none>                 7377.9 142.73
+ sex     1     219.8  7158.2 144.03
+ height  1     119.0  7258.9 144.35
+ tlc     1     115.8  7262.1 144.36
+ frc     1      33.4  7344.5 144.62
+ age     1      21.9  7356.0 144.66
- rv      1    1584.9  8962.8 145.20
- bmp     1    3355.7 10733.6 149.35
- fev1    1    4203.5 11581.4 151.10
- weight  1   12686.4 20064.4 163.74
```

```
summary(model_step)
```

```
Call:
lm(formula = pemax ~ weight + bmp + fev1 + rv, data = cystfibr_filtered)
```

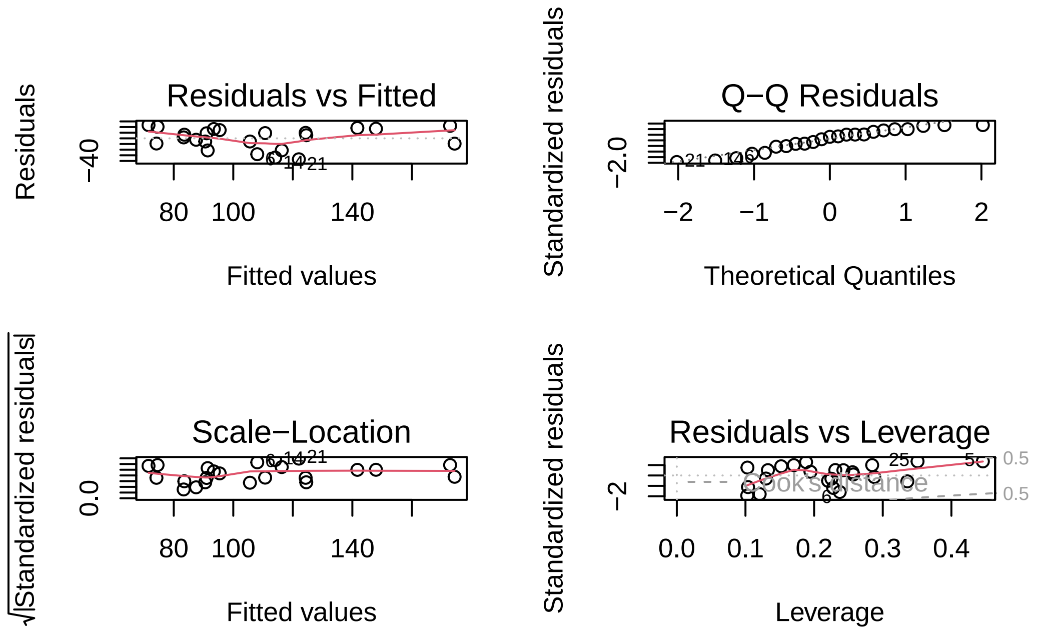
```
Residuals:
    Min      1Q  Median      3Q     Max 
-37.029  -9.267   5.475  15.550  23.467 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 56.74448   52.66854   1.077  0.29553    
weight       1.95576    0.35154   5.563 2.79e-05 ***
bmp         -1.53722    0.53725  -2.861  0.01038 *  
fev1         1.68405    0.52587   3.202  0.00494 ** 
rv           0.15430    0.07847   1.966  0.06487 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20.25 on 18 degrees of freedom
Multiple R-squared:  0.7163,    Adjusted R-squared:  0.6533 
F-statistic: 11.36 on 4 and 18 DF,  p-value: 8.856e-05
```

```r
# looking at the summary the lowest AIC is for the:
# weight + bmp + fev1+ rv model
# I can see that in that model again theR^2 is higher than before
# the residual standard error is lower than before and the P value
# very small so it is very significant

# I can also look at individual parameter P values
# so we can see that weight has the lowest one meaning that
# weight and Pemax are highly positively correlated
# then fev1 and pemax are positively correlated
# then BMP and pemax are negatively correlated
# then the P value for rv is higher than 0.05, not sure why it would not exclude it

# gonna run a test for a model without rv since its p value is higher than 0.05
model_filtered_2 <- lm(pemax ~ weight + bmp +fev1, data = cystfibr_filtered)
summary(model_filtered_2)
```

```
Call:
lm(formula = pemax ~ weight + bmp + fev1, data = cystfibr_filtered)

Residuals:
    Min      1Q  Median      3Q     Max 
```

```
-42.526 -11.017   2.331  14.089  36.673

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 139.1779    34.2049   4.069 0.000655 ***
weight        1.7363     0.3576   4.855 0.000110 ***
bmp          -1.7249     0.5672  -3.041 0.006718 **
fev1          1.1321     0.4771   2.373 0.028346 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 21.72 on 19 degrees of freedom
Multiple R-squared:  0.6554,    Adjusted R-squared:  0.601
F-statistic: 12.05 on 3 and 19 DF,  p-value: 0.0001206
```

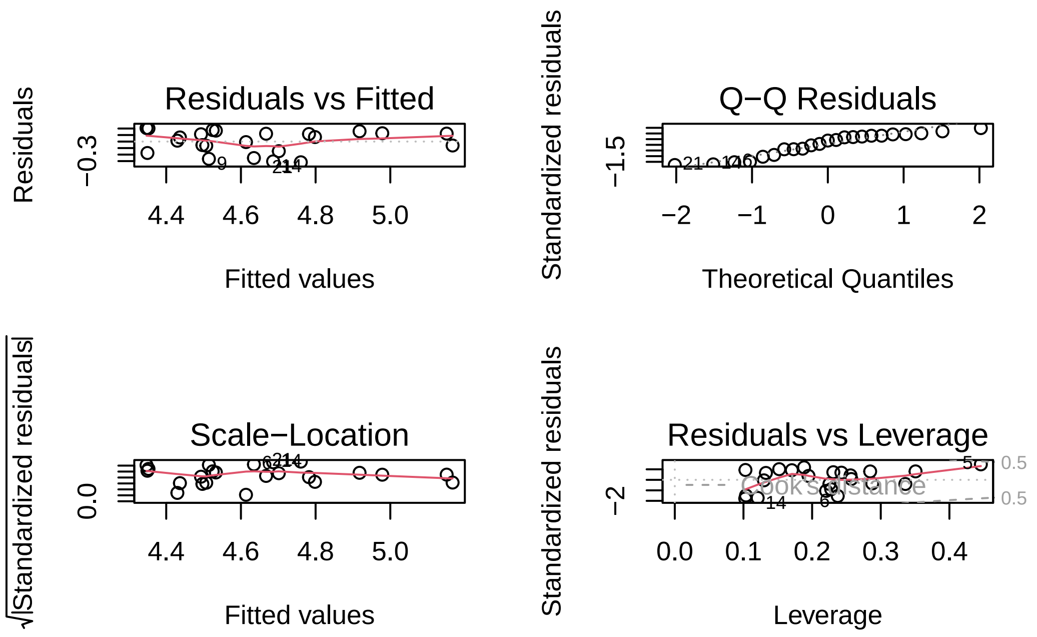
```r
# okay so the model R^2, p value and residual standard error all went in the bad way
# so gonna keep rv in there.


# now we can check the residuals
par(mfrow = c(2,2))
plot(model_step)  # Check residual plots
```



```r
# I can still see some curvature maybe I should do a log transform

cystfibr_filtered$log_pemax <- log(cystfibr_filtered$pemax)
model_log <- lm(log_pemax ~ weight + bmp + fev1 + rv, data = cystfibr_filtered)
summary(model_log)
```

```
Call:
lm(formula = log_pemax ~ weight + bmp + fev1 + rv, data = cystfibr_filtered)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.31777 -0.10607  0.06314  0.12477  0.20661 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  4.3664352  0.4901496   8.908 5.13e-08 ***
weight       0.0161132  0.0032716   4.925 0.000109 ***
bmp         -0.0136076  0.0049998  -2.722 0.013993 *  
fev1         0.0130483  0.0048939   2.666 0.015740 *  
rv           0.0010960  0.0007303   1.501 0.150728    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1884 on 18 degrees of freedom
Multiple R-squared:  0.6591,	Adjusted R-squared:  0.5833 
F-statistic: 8.699 on 4 and 18 DF,  p-value: 0.0004314
```

```r
par(mfrow = c(2,2))
plot(model_log)
```



```
# so after doing that we get a much lower residual std error
# but also a little bit lower R^2 and a little bit higher P value.
# but still the drop in residual std error is so much that I think its worth

# I think this is the best model and it tells us
# that weight, BMP, and FEV1 are highly correlated with pemax
# and you need RV to get a good fit of pemax
```

Question 5 (10 points)

Show how to parameterize the following piecewise linear spline to ensure that the resulting function is continuous:

1. Use the knots $c = 10, 30, 50, 75$ so that you will have three linear pieces.
2. Let the slope of each linear piece be free to vary then find conditions for the intercept of each linear piece to ensure the function is continuous.

- *Hint:* It might be easier to write each linear segment in the form $f_i(x) = \beta_{0,i} + \beta_{1,i}(x - l_i)$ where l_i is the left endpoint for that linear segment.

3. Modify the function below so that it takes a vector beta and an input x and evaluates the spline using your parameterization. You might need to remove β_5 and β_6 and replace them with a value using your answer to (2.)

4. Write a sum-of-squares function that takes a β parameter vector as input and gives you the sum-of-squared errors as output, using cystfibr as your dataset, pemax as your response variable, and weight as your covariate. Then use the nlminb function to find the best estimate for β and plot the estimated piecewise linear function in the same plot as the data.

So we need to think of the function like this:

$$f_i\left(x\right) = \beta_{0,i} + \beta_{1,i}\left(x - l_i\right)$$

and if we look at the knots we get these segments:

$$x \in [10, 30] \rightarrow f_1(x) = \beta_1 + \beta_2(x - 10)$$
$$x \in [30, 50] \rightarrow f_2(x) = \beta_5 + \beta_3(x - 30)$$

$$x \in [50, 75] \rightarrow f_3(x) = \beta_6 + \beta_4(x - 50)$$

To get continuity we need $f_1(30) = f_2(30)$ and $f_2(50) = f_3(50)$, so we get:

$$\beta_1 + \beta_2(30 - 10) = \beta_5 + \beta_3(30 - 30) \rightarrow \beta_1 + 20\beta_2 = \beta_5$$

$$\beta_5 + \beta_3(50 - 30) = \beta_6 + \beta_4(50 - 50) \rightarrow \beta_5 + 20\beta_3 = \beta_6$$

Then from those we can combine them to get this:

$$\beta_6 = \beta_1 + 20\beta_2 + 20\beta_3$$

we can rewrite the equations using this new information:

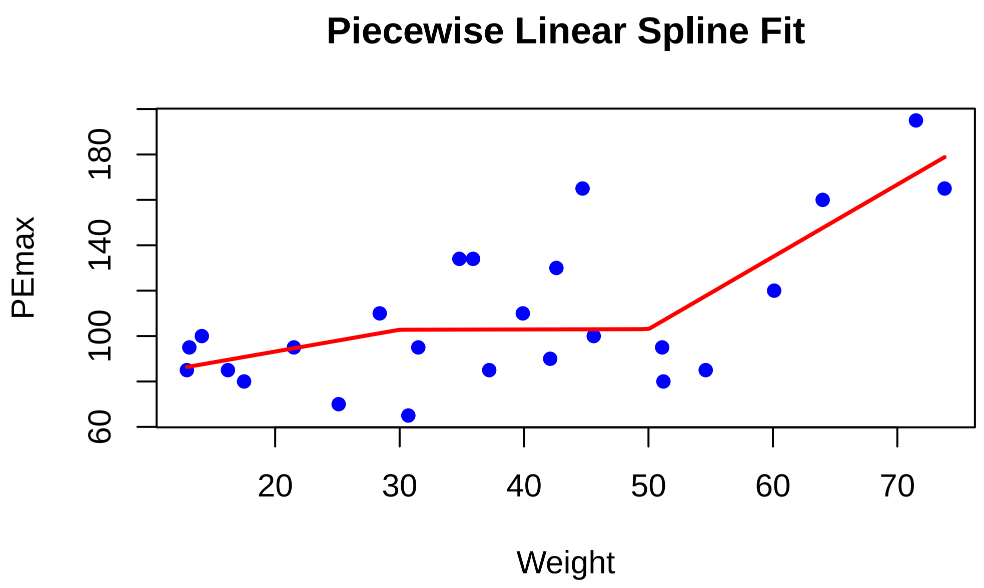
$$x \in [10, 30] \rightarrow f_1(x) = \beta_1 + \beta_2(x - 10)$$

$$x \in [30, 50] \rightarrow f_2(x) = \beta_1 + 20\beta_2 + \beta_3(x - 30)$$

$$x \in [50, 75] \rightarrow f_3(x) = \beta_1 + 20\beta_2 + 20\beta_3 + \beta_4(x - 50)$$

```r
knots<- c(10, 30, 50, 75)

f<- function(beta, x) {

    # i changed the code here cause i did not like the syntax
    which_piece<- cut(x, breaks = knots, labels = FALSE)
    linear_pieces<- list()

    # i changed the equations below to match the new ones found
    linear_pieces[[1]] <- function(x) beta[1] + beta[2] * (x - knots[1])
    linear_pieces[[2]] <- function(x) (beta[1] + 20 * beta[2]) + beta[3] * (x - knots[2])
    linear_pieces[[3]] <- function(x) (beta[1] + 20 * beta[2] + 20 * beta[3]) +
      beta[4] * (x - knots[3])

    prediction<- sapply(
      # again my syntax here is much more readable
        seq_along(x),
        function(i) linear_pieces[[which_piece[i]]](x[i])
    )
    return(prediction)
}


# my sum of squares function
sum_of_squares <- function(beta) {
    predicted <- f(beta, cystfibr$weight)
    residuals <- cystfibr$pemax - predicted
    return(sum(residuals^2))
}

# just put some random number
beta_start <- c(10, 10, 10, 10)
# how can i be sure that my starting location leads to a global minima
#and not a local minima?

# use nlminb, beta_start and to minimize the sum_of_squares function
result <- nlminb(start = beta_start, objective = sum_of_squares)

# get the best found paramters for beta
beta_best <- result$par

# now plot them
```

```r
plot(cystfibr$weight, cystfibr$pemax, pch = 16, col = "blue",
     xlab = "Weight", ylab = "PEmax", main = "Piecewise Linear Spline Fit")
curve(f(beta_best, x), add = TRUE, col = "red", lwd = 2)
```



```r
# looking at the graph i can definitely see three different slopes
# and three different segments
# so the piecewise linear function actually works
```